1/

lec 3

Yasmin samy

ch 7

sheet # 5

Figures needed for sheet 5

Solution

X $M \Rightarrow Y$ Registers

$Q \Rightarrow$ Temp

Sheet (5)

(7-9) Mul R_1, R_2

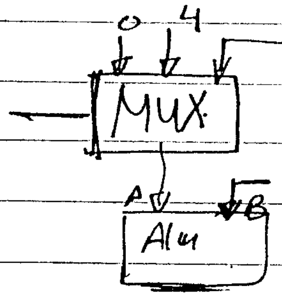
Use
- → Counter Register is available in the Bus.
- → ① write possible Control Sequence.
- → ② Use Figure 7.1 (Bus structure)
- → ③ Use Figure 6.7

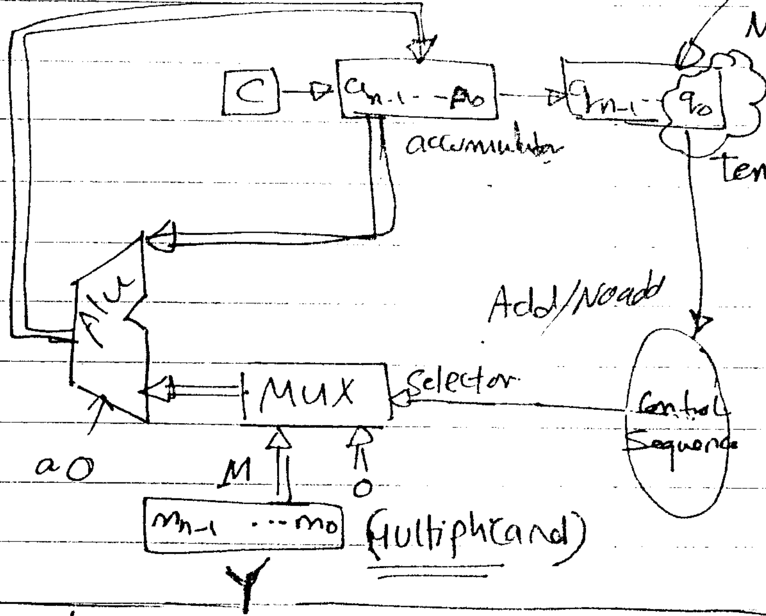
assume:-

① Z ⇒ perform the fn of the accumulator.

Temp ⇒ hold Multiplier

Y ⇒ hold Multiplicand.





Solution:

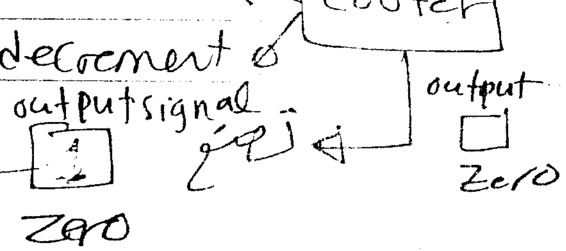
Steps	action
1.	PCout, MARin, Read, Select4, Add, Zin
2.	Zout, PCin, Yin, WMFC
3.	MDRout, IRin
4.	Constant = 32, Constantout, Counterin
5.	R_1out, TEMPin
6.	R_2out, Yin
7.	Zout, if Tempo = 1 then $\boxed{\text{SelectY}}$ else $\boxed{\text{select0}}$, Add, Zin, decrement
8.	Shift, if Zero = 0 then Branch 7
9.	Zout, R_2in, End

Fetch (steps 1–3)

q_0

لتحميل العداد بقيمة العد

[illegible]

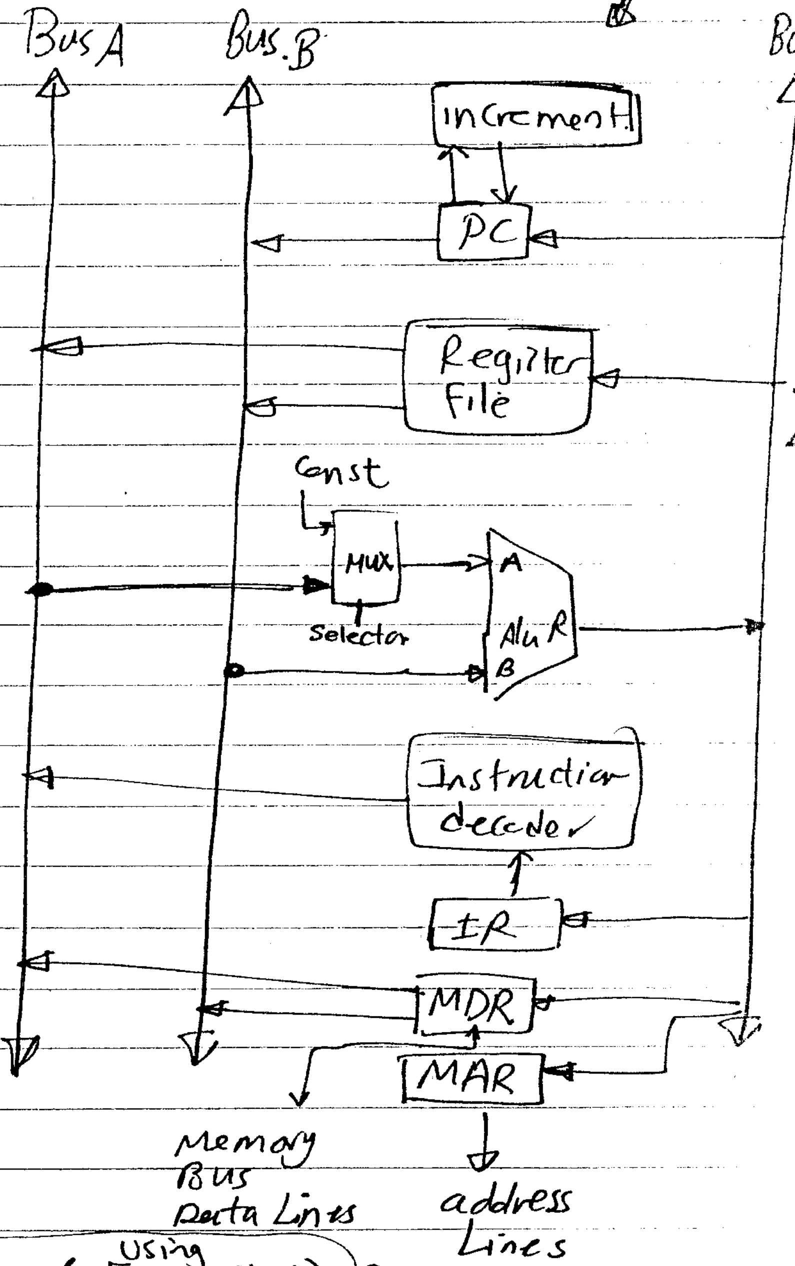


(Decrement) Zero [illegible]

Branch بجي فقط ازا كانت $N = \boxed{1}$

(7-10)

Control steps for Branch_on-Negative instruction. Figure 7.8



Solution:

1- PC_{out_B}, R=B, MAR_{in}, Read, Inc PC

2. WMFC

3. MDR_{out_B}, R=B, IR_{in}

4. PC_{out_B}, offset field of IR_{out}, Add, if N=1 then R=B, PC_{in}, END.

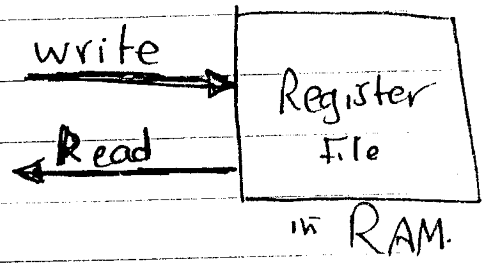
Solution (using Figure 7-1):

1–3 Fetch instruction

4 PC_{out}, offset-field of IR_{out}, Add, if N=1 then PC_{in}, End.

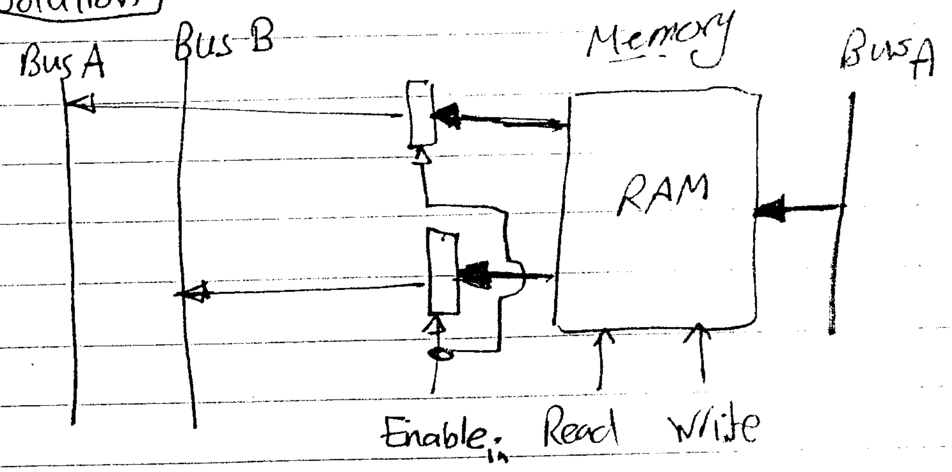
problem 7.15

Figure 7-8



Explain How you would use :
- additional Latches as either input or output of the RAM to the operate the file in master slave mode.

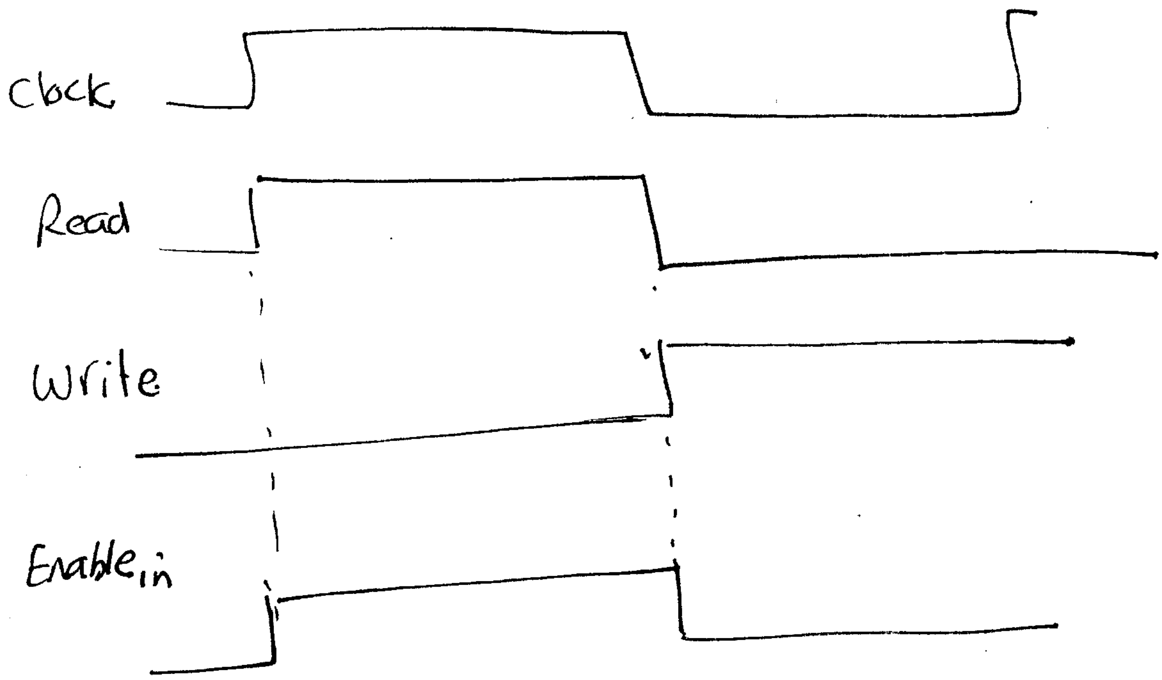
Solution



at read operation :-

first phase
① at first clock cycle. & Enable$_{in}$ = 1
② Data enter the two Latches and appears at two buses immediately.

Second phase
③ during the second phase of the clock latch inputs are disabled Enable$_{in}$ = 0
write operation → during this phase to record the result of data transfer.


Clock
Read
Write
Enable$_{in}$

Sheet #5

MOV X(Rsrc), Rdst (write microprogram)

7.24

Call microroutine

Address	Microinstruction
000	PC_{out}, MAR_{in}, Read, Select4, Add, Zin
001	Z_{out}, PC_{in}, Y_{in}, WMFC
002	MDR_{out}, IR_{in}
300	μBranch { MPC ← 161 }
161	PCout, MARin, Read, Select4, Add, Zin
162	Zout, PCin, WMFC
163	MDRout, Yin
164	$R_{src\,out}$, Select Y, Add, Zin
165	Zout, MARin, Read
166	μBranch { MPC ← 170; MPC_0 ← [IR_8] }, WMFC
170	MDRout, MARin, Read, WMFC
171	MDRout, Yin
172	$Rdst_{out}$, Select Y, Add, Zin
173	Zout, Rdst, End.

Same as figure 7.21 (000–300)

X (161–163)

R_{src} (164)

Same as Fig 7.21 (170–173)

حتى نستطيع كتابة هذا البرنامج بهذه الكيفية لابد
من النظر إلى Figure 7-20

Figures needed to solve
Sheet #5

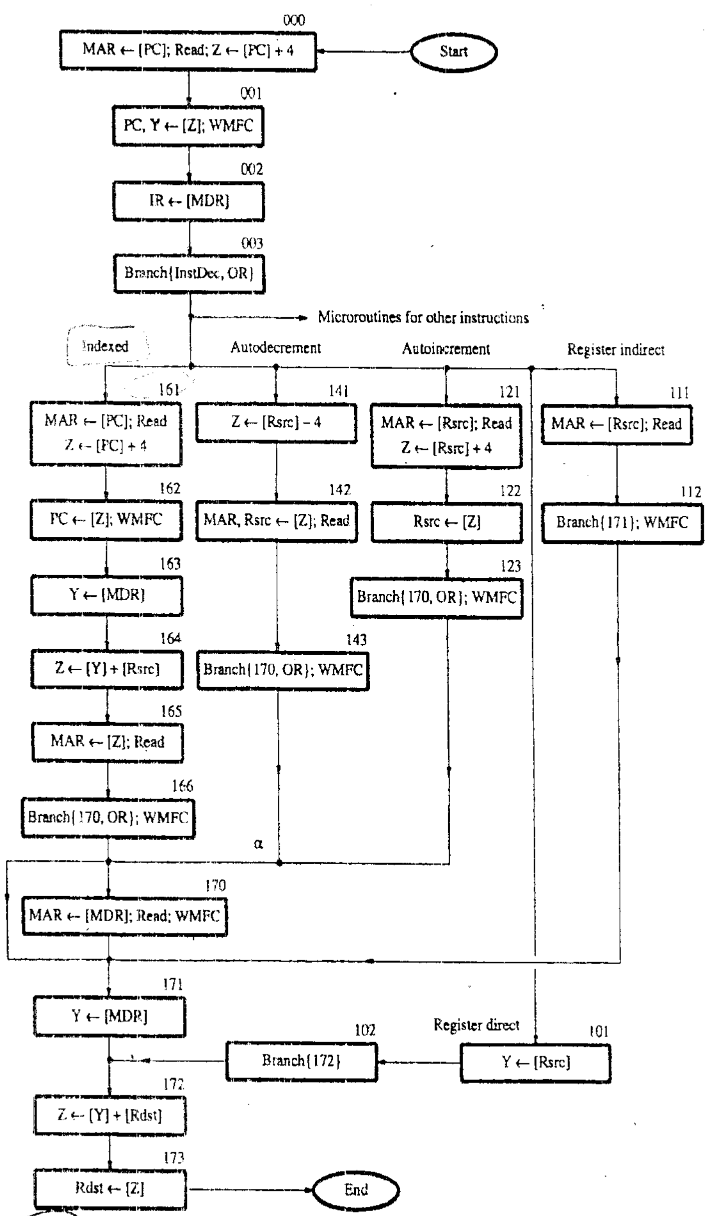


Figure 7.20 Flowchart of a microprogram for the Add src,Rdst instruction.

problem 7.24

Mov $X(R_{src}), R_{dst}$

1. PCout, MARin, Read, Select4, Add, Zin
2. Zout, PCin, Yin, WMFC.
3. MDRout, IRin
4. offset-field-of-IRout, MARin, Read
5. Rsrcout, Yin, WMFC
6. MDRout, SelectY, Add, Zin
7. Zout, MARin, Read.
8. WMFC
9. MDRout, Rdst in, End

Branch-on-Negative
Branch<0 loop

if N=0 No Branch
if N=1 Branch

(Single Bus Structure)

1. PCout, MAR_{in}, Read, Select4, Add, Z_{in}
2. Zout, PC_{in}, Y_{in}, WMFC
3. MDRout, IR_{in}
4. Offset-field-of-IRout, Select Y, Add, Zin,
   if N=0 then End
5. Zout, PC_{in}, End.

(Multiple-Bus Structure)

1. $PCout_B$, R=B, MAR_{in}, Read, IncPC
2. WMFC
3. $MDRout_B$, R=B, IR_{in}
4. $PCout_B$, Offset-field-of-$IRout_A$, Add,
   if N=0 then End
5. PC_{in}, End

Z → Accumulator (A)

Temp → Q

Y → M

Mul R_1, R_2

Q M

Temp Y

assume we can make Zout, Zin

M → Y

Q → Temp → $Temp_0$ (q_0)

ACC

counter

Zeros

0 1

Counter لم يخلص

وصلنا لل
counter خلص

1. PCout, MAR_{in}, Read, Select4, Add, Zin
2. Zout, PC_{in}, Y_{in}, WMFC
3. MDRout, IR_{in}
4. Constant = 32, $Constant_{out}$, $Counter_{in}$ فرض

5. R_1out, $Temp_{in}$ Q
6. R_2out, Y_{in} M

accumulator $q_0 = 1$ M 0

7. Zout if $Temp_0 = 1$ then SelectY else Select0,

Add, Z_{in}, decrement العداد

طول ما هناك أي (bit) أخرى في Q

8. shift, if Zero = 0 then Branch 7

9. Zout, R_{2in}, End

(7.30) @ Assi

Address	Microinstruction
0000	A
0001	B
0010	if ($(b_6 b_5) = 00$) then µBranch 0111
0011	if ($(b_6 b_5) = 01$) then µBranch 1010
0100	if ($(b_6 b_5) = 10$) then µBranch 1100
0101	I
0110	µBranch 1111
0111	C
1000	D
1001	µBranch 1111
1010	E
1011	µBranch 1111
1100	F
1101	G
1110	H
1111	J

(b) Assume that bits b_{6-5} of IR are ORed into bit MPC_{3-2}

Address	Microinstruction
0000	A
0001	B; $MPC_{3-2} \xleftarrow{OR} b_{6-5}$
$\boxed{0010}$	C
0011	D
0100	μBranch $\boxed{1111}$
$\boxed{0101}$	E
0110	μBranch $\boxed{1111}$
$\boxed{0111}$	F
1011	G
1100	H
1101	μBranch $\boxed{1111}$
$\boxed{1110}$	I
$\boxed{1111}$	J

$$MPC_{3-2} \xleftarrow{OR} b_{6-5}$$

ناتج OR لازم هو الذي يعطي المكان الذي سوف ننتقل إليه،
وهو إما
$\boxed{0010}$ أو $\boxed{0101}$
أو $\boxed{0111}$
أو $\boxed{1110}$

(c)

Address	Microinstruction: Next address	Microinstruction: Function
0000	0001	A
0001	0010	B; $MPC_{3-2} \leftarrow b_{6-5}$
0010	0011	C
0011	$\boxed{1111}$	D
0110	$\boxed{1111}$	E
1010	1011	F
1011	1100	G
1100	$\boxed{1111}$	H
1110	$\boxed{1111}$	I
$\boxed{1111}$	—	J

Figure needed for ~~Problem 7.32~~
Figure 7.19 & 7-20
Example Figure 7.6
Figure 7.7

Figure 7.19
(1) 7-31
(2) 7.32
(3) 7.33

Microinstruction

F1	F2	F3	F4	F5

F1 (4 bits)	F2 (3 bits)	F3 (3 bits)	F4 (4 bits)	F5 (2 bits)
0000: No transfer	000: No transfer	000: No transfer	0000: Add	00: No action
0001: PC_{out}	001: PC_{in}	001: MAR_{in}	0001: Sub	01: Read
0010: MDR_{out}	010: IR_{in}	010: MDR_{in}	⋮	10: Write
0011: Z_{out}	011: Z_{in}	011: $TEMP_{in}$		
0100: $R0_{out}$	100: $R0_{in}$	100: Y_{in}	1111: XOR	
0101: $R1_{out}$	101: $R1_{in}$		16 ALU functions	
0110: $R2_{out}$	110: $R2_{in}$			
0111: $R3_{out}$	111: $R3_{in}$			
1010: $TEMP_{out}$				
1011: $Offset_{out}$				

problem 7-31

problem 7-31

F6	F7	F8	...

F6 (1 bit)	F7 (1 bit)	F8 (1 bit)
0: SelectY	0: No action	0: Continue
1: Select4	1: WMFC	1: End

Figure 7.19 An example of a partial format for field-encoded microinstructions.

then be assigned a distinct code that represents the microinstruction. Such full encoding is likely to further reduce the length of microwords but also to increase the complexity of the required decoder circuits.

Highly encoded schemes that use compact codes to specify only a small number of control functions in each microinstruction are referred to as a *vertical organization.* On the other hand, the minimally encoded scheme of Figure 7.15, in which many resources can be controlled with a single microinstruction, is called a *horizontal organization.* The horizontal approach is useful when a higher operating speed is desired and when the machine structure allows parallel use of resources. The vertical approach results in considerably slower operating speeds because more microinstructions are needed to perform the desired control functions. Although fewer bits are required for each microinstruction, this does not imply that the total number of bits in the control store is smaller. The significant factor is that less hardware is needed to handle the execution of microinstructions.

(1) **(7-31)** need figure 7-19.

Sol/n: To reduce the number of bits needed to encode control signals in Figure (7-19) we do the following:

① put the Y_{in} Control Signal as the fourth Signal in F_5 to reduce F_3 by one bit

② Combine fields F_6, F_7, and F_8 into a single 2-bit field that represents:

00 : Select4
01 : Selecty
10 : WMFC
11 : END

finally

القديمة F_6 F_7 F_8

F_1 (4 bits)	F_2 (3 bits)	F_3 (2 bits)	F_4 (4 bits)	F_5 (2 bits)	F_6 (2-bits)
the same as Figure 7-19	the same as Figure 7-19	00: No transfer 01: MARin 10: MDRin 11: Tempin تم ازاحة Y_{in} من هنا ووضعها في F5	The same as Figure 7-19	00: No action 01: Read 10: Write 11: Yin	00 : Select 4 01 : Selecty 10 : WMFC 11 : End- تم تصغيرها للمستوى عن الموجود من F6 & F7 & F8

Need Figure 7.19, Figure 7.6, Figure 7.7

(7.32) Suggest a new encoding scheme to generate control signals of figure (7.19)

- that reduce number of bits needed to **12**.
- explain your effect to examples figure 7.6 & figure 7.7

Solution

هنا سوف نقوم بدلاً من استخدام وتوليد كل (Control Signal) على حدى نقوم بالمرور على جميع الإشارات التحكم الموجودة ومعرفة أى إشارات مرتبطة وجودها مع بعضها فنكتب مثلاً:

1. MAR_{in} & Read $\Longrightarrow$ $MAR_{in} \cdot Read$ (and.)
2. Z_{in} → Select4, SelectY $\Longrightarrow$ $Z_{in} \cdot Select4$, $Z_{in} \cdot SelectY$

new encoding scheme will be:

FA: $F_1 + Z_{out} \cdot End$, $Z_{out} \cdot WMFC$ (11 signal)

FB: $F_2, F_3 \xrightarrow{+}$ $Z_{in} \cdot Select4$, $Z_{in} \cdot SelectY$, MAR_{in}, $MAR_{in} \cdot Read$, $MDR \cdot write$ } مدمجة: Z_{in}, MAR_{in}, MDR_{in} (13 signals)

FC: F4 (16 signals)

(10) بعد هذا التعديل سوف نقوم بتطبيق أثره على الأمثلة الموجودة في Figure 7.7 & Figure 7.6

سيحدث تعديل في خطوة (5) كما في الصفحة القادمة ← لم يحدث تعديل

the contents of R1 are transferred to register Y in step 5, to prepare for the addition operation. When the Read operation is completed, the memory operand is available in register MDR, and the addition operation is performed in step 6. The contents of MDR are gated to the bus, and thus also to the B input of the ALU, and register Y is selected as the second input to the ALU by choosing SelectY. The sum is stored in register Z, then transferred to R1 in step 7. The End signal causes a new instruction fetch cycle to begin by returning to step 1.

This discussion accounts for all control signals in Figure 7.6 except Y_{in} in step 2. There is no need to copy the updated contents of PC into register Y when executing the Add instruction. But, in Branch instructions the updated value of the PC is needed to compute the Branch target address. To speed up the execution of Branch instructions, this value is copied into register Y in step 2. Since step 2 is part of the fetch phase, the same action will be performed for all instructions. This does not cause any harm because register Y is not used for any other purpose at that time.

7.2.1 BRANCH INSTRUCTIONS

A branch instruction replaces the contents of the PC with the branch target address. This address is usually obtained by adding an offset X, which is given in the branch instruction, to the updated value of the PC. Figure 7.7 gives a control sequence that implements an unconditional branch instruction. Processing starts, as usual, with the fetch phase. This phase ends when the instruction is loaded into the IR in step 3. The offset value is extracted from the IR by the instruction decoding circuit, which will also perform sign extension if required. Since the value of the updated PC is already available in register Y, the offset X is gated onto the bus in step 4, and an addition operation is performed. The result, which is the branch target address, is loaded into the PC in step 5.

The offset X used in a branch instruction is usually the difference between the branch target address and the address immediately following the branch instruction.

problem 7-32

Figure 7.7

[illegible] structure [illegible]

Step	Action
1	PC_{out}, MAR_{in}, Read, Select4, Add, Z_{in}
2	Z_{out}, PC_{in}, Y_{in}, WMFC
3	MDR_{out}, IR_{in}
4	Offset-field-of-IR_{out}, Add, Z_{in}
5	Z_{out}, PC_{in}, End

Figure 7.7 Control sequence for an unconditional Branch instruction.

7.2 Execution of a Complete Instruction

Let us now put together the sequence of elementary operations required to execute one instruction. Consider the instruction

Add (R3),R1

which adds the contents of a memory location pointed to by R3 to register R1. Executing this instruction requires the following actions:

1. Fetch the instruction.
2. Fetch the first operand (the contents of the memory location pointed to by R3).
3. Perform the addition.
4. Load the result into R1.

Figure 7.6 gives the sequence of control steps required to perform these operations for the single-bus architecture of Figure 7.1. Instruction execution proceeds as follows. In step 1, the instruction fetch operation is initiated by loading the contents of the PC into the MAR and sending a Read request to the memory. The Select signal is set to Select4, which causes the multiplexer MUX to select the constant 4. This value is added to the operand at input B, which is the contents of the PC, and the result is stored in register Z. The updated value is moved from register Z back into the PC during step 2, while waiting for the memory to respond. In step 3, the word fetched from the memory is loaded into the IR.

Steps 1 through 3 constitute the instruction fetch phase, which is the same for all instructions. The instruction decoding circuit interprets the contents of the IR at the beginning of step 4. This enables the control circuitry to activate the control signals for steps 4 through 7, which constitute the execution phase. The contents of register R3 are transferred to the MAR in step 4, and a memory read operation is initiated. Then

لأنه 2 فيه [illegible]

5. Zout, WMFC
6. R1out, Yin
7. MDRout, selectY, Add, Zin
8. Zout, R1in, End

Step	Action
1	PC_{out}, MAR_{in}, Read, Select4, Add, Z_{in}
2	Z_{out}, PC_{in}, Y_{in}, WMFC
3	MDR_{out}, IR_{in}
4	$R3_{out}$, MAR_{in}, Read
5	$R1_{out}$, Y_{in}, WMFC
6	MDR_{out}, SelectY, Add, Z_{in}
7	Z_{out}, $R1_{in}$, End

Figure 7.6 Control sequence for execution of the instruction Add (R3),R1.

Figure 7.6 problem 7.3

سوف نعمل الخطوة 5 و 6
لا نستطيع (5 و 6)
لأنه يصير فيه 2 output
Zout, R1in
في نفس الوقت 2

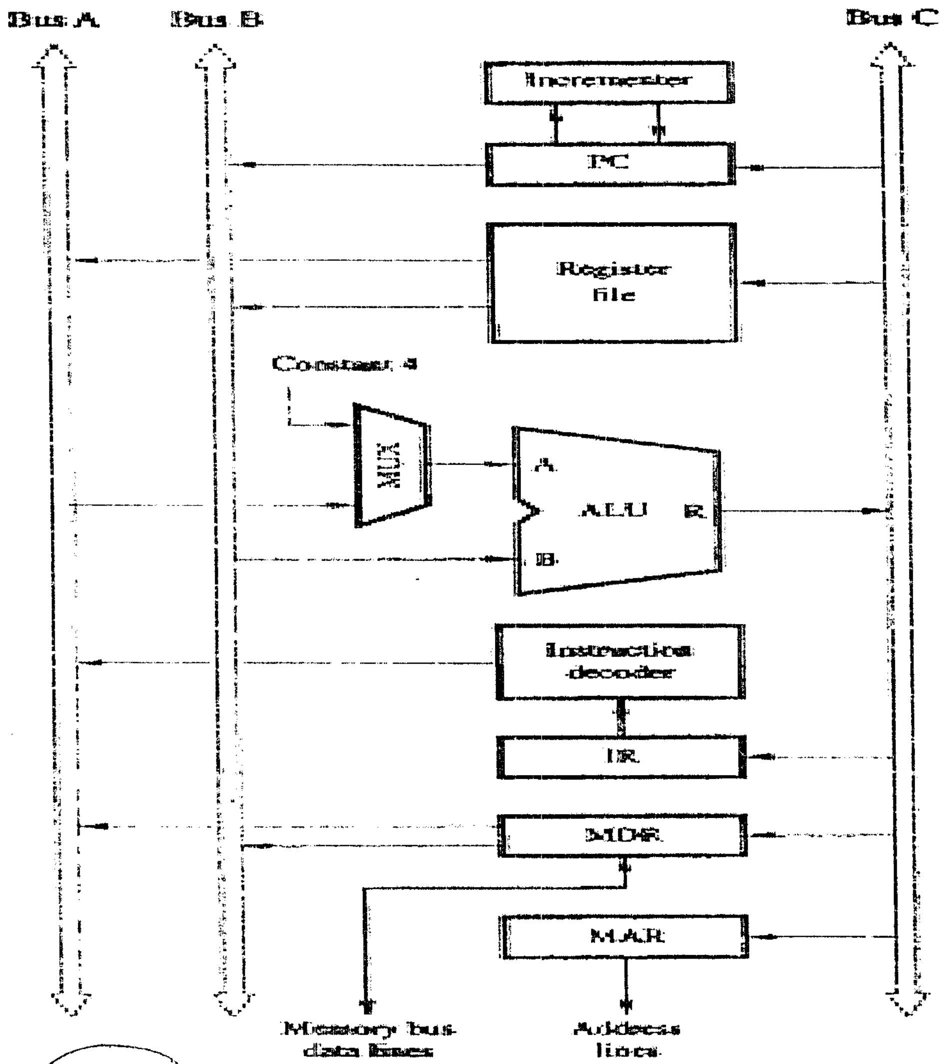


Figure 7.8 Three-bus organization of the datapath.

we need figure 7.19 & figure 7-8

7-33 suggest a formate for microinstruction simillar to figure 7.19. if the processor Organization as shown in figure (7-8)

Solution :

- Figure 7-8 contain 2 Bus A & B connected to the Alu.

∴ 2 fields are needed instead of F_1

لـ تحتوي على جميع إشارات out مثل MDRout

سوف نستبدل بعمودين MDRoutA, MDRoutB كالآتي:

F_1_A	F_1_B
MDRoutA	MDRoutB
$R_i$outA	$R_j$outB
:	:

7-34 merits of horizontal and vertical microinst.

horizontal microinstructions	vertical microinstructions
• They are Longer • They need a Large microprogram area • used for CISC	• They are require more encoding and decoding of signals • Longer delay. • lead to Longer microprograms and slower operation.

(7-35)

(a) Advantages of Hardwired Control:

- It is fast operation.

Disadvantages of Hardwired Control:

- Higher Cost.
- عدم مرونة
  Inflexibility when changes or additions are to be made.
- Longer time required to design and implement such Unit.

(b) advantages of microprogrammed Control:

- low cost
- high flexability when changes are made.

disadvantages of microprogramed Control:

- lower speed of operation. becomes a problem in high performance computer.